HOCHSCHULE BONN-RHEIN-SIEG

Software Development Project
Project Descriptions

SUMMER SEMESTER 2018

This document describes projects for the Software Development Project course in SS2018. The descriptions here are meant as a starting point; you can run a working version of the described components by following the instructions. The requirements and proposed solutions of the projects are subject to change during the course of the semester as you get familiar with the code and are able to propose your own solutions.

During the course of the project, you will refactor and/or rewrite code, but the basic behaviour of the components should remain the same.

1 Setup

This section describes the initial setup needed for all projects. You will clone and compile the code required for your project.

1. Remove any sourcing of existing catkin workspaces from your .bashrc, and restart the terminal
2. Create a new catkin workspace (for example: `mkdir -p sdp_ws/src`)
3. Clone the repositories needed for your project into the catkin workspace.
4. Install dependencies (as specified in your repository)
5. Use `catkin build` to compile the workspace

2 Projects

2.1 Project 1: Object List Merger

The object list merger (OLM) is a ROS node that merges two or more `ObjectList` messages and publishes the merged `ObjectList`. The input `ObjectLists` are generated by different object detection and recognition pipelines, but are based on the same scene. In the current setup, one `ObjectList` is generated by the perception pipeline which uses 3D pointclouds, and the second `ObjectList` is generated by the 2D perception pipeline. If available, the OLM also uses prior knowledge about the location of objects (from the knowledge base) to improve the results.

2.1.1 Problem

The code for this component is poorly written. Some of the issues include:

- the code is not readable (i.e. it is difficult to understand what's going on without having additional knowledge that's not in the code)
- configuration parameters / prior knowledge about specific objects is included in the code
- the code is tailored to the two object detection/recognition methods that are currently being used
- some debug messages are written to a file instead of the terminal

2.1.2 Proposed Solution

Refactor the code such that the above problems are addressed. In particular

- the code should be readable, with sufficient, but not excessive, documentation. In the ideal world, the code *is* the documentation.
- all prior knowledge, or parameters specific to objects or the object recognition method used must be configurable. The code should be as general as possible, and any tailored items must be put into configuration files.
- write test cases that ensure the program performs as expected

2.1.3 Data

Initially, you will use data that has been previously recorded on the youBot in the @Work lab. The data consists of a set of bagfiles as follows: `nn_refbox.bag`: contains order and inventory messages from the referee box for set `nn`. `nn_object_list_mm.bag`: contains the recognized object list messages for set `nn`. There are up to three object list bagfiles for each set. These correspond to different runs of object recognition, and is denoted by `mm`.

Each set also corresponds to different locations.

- Set 01: WS02

- Set 02: WS09
- Set 03: WS04
- Set 04: WS05

You can find the bagfiles here:
`https://drive.google.com/open?id=1KwzSwAnpZ6uiSs3ZswtLwSzhjOdWYQW6`

2.1.4 Launch

```
roscore
roslaunch mir_refbox_parser refbox_parser.launch
roslaunch mcr_pddl_problem_generator rosplan_knowledge_base_example.launch
roslaunch mir_knowledge_base_analyzer knowledge_base_queries.launch
roslaunch mir_knowledge upload_yb_intrinsic_knowledge.launch
roslaunch mcr_object_list_merger object_list_merger.launch
rosrun rqt_gui rqt_gui
```

- `refbox parser`: uploads facts and goals to the knowledge base based on the inventory and order specified by the referee box
- `rosplan knowledge base`: Mongodb knowledge base
- `knowledge base queries`: component that queries the KB (in this case the only query used is `get_objects_at_current_location`)
- `upload yb intrinsic knowledge`: uploads additional facts not specified by the referee box (for example: the robot is at `START`)
- `rqt gui`: for visualizing the knowledge base (add Plugins->ROSPlan->ROSPlan dispatcher)

2.1.5 Run

```
rosbag play nn_refbox.bag
rostopic pub /upload_knowledge/event_in std_msgs/String e_trigger
rosrun mir_knowledge_base_analyzer set_robot_position WSXX
rostopic pub /mcr_perception/object_list_merger/event_in std_msgs/String e_start
rosbag play nn_object_list_mm.bag
rostopic echo /mcr_perception/object_list_merger/output_object_list
rostopic pub /mcr_perception/object_list_merger/event_in std_msgs/String e_trigger
```

- `nn refbox.bag`: ROS bag file which has recorded inventory and order messages from the referee box
- `upload knowledge event`: trigger upload of intrinsic knowledge to KB
- `set robot position`: update the knowledge base with the current position of the robot (replace XX with the workstation number corresponding to the bagfile you played).

- `object list merger start event`: reset object list merger
- `nn object list mm.bag`: ROS bag file which has recorded object list messages from two object recognition implementations
- `output object list`: the merged object list topic
- `object list merger trigger event`: publish the merged object list

2.2 Project 2: Task Parser

The task parser (also called referee box parser), parses messages received from a referee box, and inserts items into the knowledge base of the robot. Two types of messages are received: Inventory and Orders. Inventory messages consist of objects and their locations. Order messages consist of objects and their target locations. Once these items are inserted into the knowledge base in a specific format, a task planner is able to generate a plan to fulfill the order given the current inventory.

2.2.1 Problem

- hard for a new comer to make changes
- has code that does not belong there
- no tests

2.2.2 Proposed Solution

- refactor
- write tests

2.2.3 Data

Your data consists of a set of bagfiles which contain inventory and order messages from the referee box. Each bagfile corresponds to a different set of objects and locations. Before you begin making changes to the code, take a note of the goals and facts uploaded to the knowledge base from each of the bagfiles. This will be your ground truth for writing tests later.

You can find the bagfiles here:
`https://drive.google.com/open?id=1Y5DWmYvKroE1uOROXKOlnbteAu8aYZes`

2.2.4 Launch

```
roscore
roslaunch mir_refbox_parser refbox_parser.launch
roslaunch mcr_pddl_problem_generator rosplan_knowledge_base_example.launch
rosrun rqt_gui rqt_gui
```

- `rosplan knowledge base`: Mongodb knowledge base
- `rqt gui`: for visualizing the knowledge base (add Plugins->ROSPlan->ROSPlan dispatcher)

2.2.5 Run

```
rosbag play nn_refbox.bag
```

`nn refbox.bag`: ROS bag file which has recorded inventory and order messages from the referee box

2.3 Project 3: 3D object detection and tracking

The Rotating Turntable Test (RTT) in the RoboCup@Work competition[1] requires the detection, tracking and recognition of objects moving on a turntable. An existing `scene_segmentation` component segments objects lying on a planar surface using 3D pointclouds. In order to use this component for RTT, some refactoring and new functionality is required.

2.3.1 Problem

The following are not problems, but are instead missing requirements in the existing `scene_segmentation` component that would make it suitable for the RTT:

- in it's current set up, it is not programmed for dynamic scenes (i.e. it performs a one-shot segmentation on a static scene)
- has no tracking component
- has nothing that exploits the "circular" nature of the motion of the objects
- cannot estimate speed and motion path of the segmented objects

2.3.2 Proposed Solution

Program the missing requirements, including:

- refactor to account for the dynamic scene. In particular, the plane needs to be extracted just once, while multiple segmentation and clustering operations can be performed using the once extracted plane.
- track objects using a naive nearest-neighbour approach
- fit a circle to the tracked objects and estimate their speed and position on the table
- output an estimated time when an object will be in front of the robot for grasping

2.3.3 Data

Test data is provided in the form of ROS bagfiles. The bagfiles contain a pointcloud topic, and the TF topic.

You can find the bagfiles here:
`https://drive.google.com/open?id=1n5wKlPvax5ijAfmOimKNnlf5-JAK8FS5`

[1] http://www.robocupatwork.org/download/rulebook-2018-01-30.pdf

2.3.4 Launch

```
roscore
roslaunch mcr_scene_segmentation scene_segmentation.launch
rviz
rosrun rqt_reconfigure rqt_reconfigure (optional)
```

Configure `rviz` as follows:

- set the global frame to `odom`
- Subscribe to `/mcr_perception/scene_segmentation/labels`
- Subscribe to `/mcr_perception/scene_segmentation/bounding_boxes`
- Subscribe to `/mcr_perception/scene_segmentation/tabletop_clusters`
- Subscribe to `/arm_cam3d/depth_registered/points`

You might want to use `rqt_reconfigure` to increase the range of `passthrough_x`.

2.3.5 Run

```
rosbag play -l X.bag
rostopic echo /mcr_perception/object_detector/object_list
rostopic pub /mcr_perception/scene_segmentation/event_in std_msgs/String e_start
rostopic pub /mcr_perception/scene_segmentation/event_in std_msgs/String e_add_cloud_start
rostopic pub /mcr_perception/scene_segmentation/event_in std_msgs/String e_segment
```

To restart, first publish `e_reset`, then the same sequence of events above.

- the `-l` option plays the bagfile in a loop
- `object_list`: the output `ObjectList` message containing segmented objects
- `e_start` start the component
- `e_add_cloud_start` add one pointcloud to the octomap
- `e_segment` segments the octomap
- `e_reset` resets the octomap

2.4 Project 4: Arm control

A common task in robot manipulation is to move the end-effector of a manipulator in a specified path (for example, to weld or paint a part). The code in the package `mir_rockin_control_fbm` was developed for a competition 3 years ago; the task was to either follow (with the end-effector) a straight line of a given slope, or a sine wave path with a given frequency and amplitude.

2.4.1 Problem

The code does not work (at least in simulation). It is also not easily transferable to other (similar tasks). For example, a new task is to follow a path which is not pre-specified, but instead on which is drawn on a sheet of paper. Hence the task would be to perceive and follow the path simultaneously. The existing code works only if the path is fully known beforehand.

2.4.2 Proposed Solution

- The non-working code has been narrowed down to `mcr_trajectory_time_parameterizer`. Debug and make the component run.
- Refactor the code such that arm trajectories can be generated and executed as new information (about the path) is available.

2.4.3 Setup

```
git clone https://github.com/b-it-bots/mas_industrial_robotics.git
```

Follow the remaining instructions from the README of the repository

2.4.4 Launch

```
roscore
roslaunch mir_bringup_sim robot.launch
rosrun gazebo_ros gzclient (only if you want to view the robot in gazebo)
roslaunch mir_moveit_youbot_brsu_1 move_group.launch
roslaunch mir_rockin_control_fbm_launcher rockin_control_fbm.launch
rviz
```

Minimal rviz configuration:

- set the global frame to `odom`
- add RobotModel
- add TF

Add other topics as necessary.

2.4.5 Run

Send the arm to a preconfigured pose:

```
rosrun moveit_commander moveit_commander_cmdline.py
use arm_1
c=[1.0 1.78146637708 -1.68232357583 3.40582651518 1.57]
go c
```

This arm configuration can be saved in `mir_moveit_youbot_brsu_1/config/youbot.srdf`, so you don't have to type it in every time.

The following commands are also in the README.md of the package mir_rockin_control_fbm_launcher.

```
rostopic pub /compute_transform/reference_point geometry_msgs/PointStamped "header:
  seq: 0
  stamp:
    secs: 0
    nsecs: 0
  frame_id: 'task_link'
point:
  x: 0.0
  y: 0.0
  z: 0.0"

rostopic pub /compute_transform/event_in std_msgs/String "data: 'e_start'"

rostopic pub /path_generator/start_point geometry_msgs/PointStamped "header:
  seq: 0
  stamp:
    secs: 0
    nsecs: 0
  frame_id: 'task_link'
point:
  x: 0.0
  y: 0.0
  z: 0.0"

rostopic pub /path_generator/end_point geometry_msgs/PointStamped "header:
  seq: 0
  stamp:
    secs: 0
    nsecs: 0
  frame_id: 'task_link'
point:
  x: 0.1
  y: 0.1
  z: 0.0"

rostopic pub /linear_interpolator_demo/event_in std_msgs/String "data: 'e_start'"
rostopic pub /linear_interpolator_demo_trajectory_executor/event_in std_msgs/String
 "data: 'e_start'"
```

2.5 Project 5: Sensor-based fault detection and diagnosis (SFDD)

This project is based on the master's thesis of Saugata Biswas. The master's thesis dealt with a sensor-based fault detection and diagnosis method proposed by Khalastchi et al.[1]. The main idea is to use correlation relationships between different variables to detect when a fault occurs. The code from his project implements the SFDD method in Python.

2.5.1 Problem

The code is not yet ready to be used on a robot because:

- no direct interface to data from a robot (for example, a ROS wrapper)
- lots of parameters (which ones are important? what are ideal values for them)
- the code was written as a proof-of-concept to show results with pre-recorded data

2.5.2 Proposed Solution

- Refactor / rewrite it with an implementation on a robot in mind
- Possibly port it to `C++` for better performance
- Write a ROS wrapper for getting data from a running robot (as opposed to log files)
- Two versions of the software exist: the original SFDD method and a modified SFDD method. Both versions need to be part of the same code base, with an option to switch between the two.

2.5.3 Resources

The master's thesis of Saugata Biswas is available on LEA.

2.5.4 Run

Run the main file in either `basic_sfdd` or `modified_sfdd` `python main.py`

[1] Eliahu Khalastchi, Meir Kalech, and Lior Rokach. Sensor fault detection and diagnosis for autonomous systems. In Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems, pages 1522. International Foundation for Autonomous Agents and Multiagent Systems, 2013.